United States Department of Agriculture

E-Gov Travel Pegasys 7.8

Operator's Guide

Contract: HHSN316201200011W

Order: AG-3144-D-16-0278

Version 4.00

April 2021

Table of Contents

Table of Figures

Table of Tables

Revision Log

Date	Version No.	Description	Author	Reviewer	Review Date
2/11	.1	Original Draft	Danielle Becker	Todd Albrigo Geoff Schutta	1/11
4/11	1.0	Final Version		Jennifer Ritchey	4/11
1/18	1.0	Updated to reflect the following changes for Pegasys 7.5.1 Upgrade: • Applied 508 standards and performed testing. • Updated screen shots and provided Alternate Text."	Jason Pfaff	Marice Grissom Jenny Lewis Sidney Ward	9/17
5/18	2.0	• Final Version 2.0 - All CWGT references have been updated to "Concur." • All E2 system references have been updated to "ConcurGov."	Jason Pfaff, Peter Timmins	Jenny Lewis, Marice Grissom	5/18
1/21	3.0	Updated to reflect the following changes for Pegasys 7.8 Upgrade: • Applied 508 standards and performed testing. • Corrected spelling of Figure 9 in Table of Tables and on Page 17. • Changed Internet Explorer to Google Chrome for functionality on Page 22. • Replaced all instances of MF_Notification with ETRAV_NotifIn, which is the queue now dedicated to this integration. • Updated screen shots/alt text as needed.	Colleen Carnes	Joseph Elliott, Marice Grissom, Jon Lavallee	1/21

Date	Version No.	Description	Author	Reviewer	Review Date
4/21	4.0	Updated to reflect the following changes for Pegasys 7.8 Upgrade • Applied USDA/GSA comments dated 2/21.	Colleen Carnes	Marice Grissom	2/21

1 Introduction

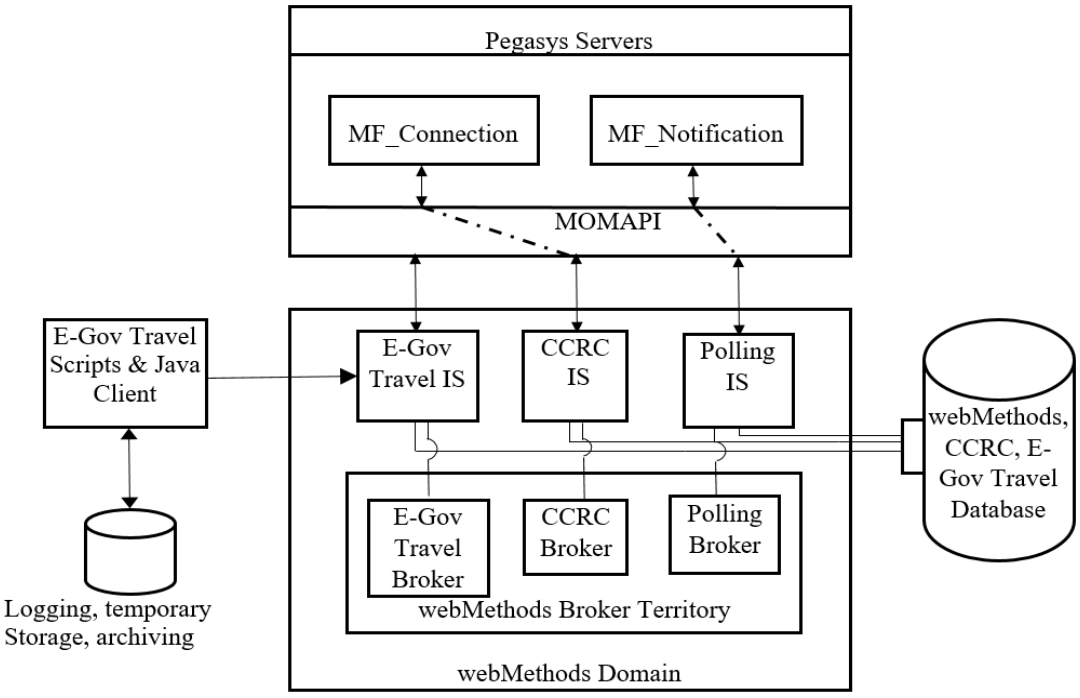
The E-Gov Travel integration interface is a collection of hardware and software components that collaborate to enable the end-to-end transmission of travel related information (travel vouchers, acknowledgements and disbursement notifications) between ConcurGov and GSA's implementation of the CGI Momentum® Financials product (known throughout GSA as Pegasys).

1.1 **Document Overview**

The E-Gov Travel Operator's Guide describes aspects of the installation and configuration of custom software specific to this integration, such as E-Gov Travel webMethods packages and Secure FTP Perl scripts used for file transfers. It also contains sections with information regarding configuration changes required for Pegasys and steps for troubleshooting the integration interface.

The remainder of this document is broken down into sections based on the physical separation of responsibilities as depicted in **Figure 1**.

Figure 1: Pegasys Servers



- **Section 2** discusses installation, operation and troubleshooting procedures for the eTravel file transfer scripts.
- **Section 3** discusses E-Gov Travel specific configuration items.
- **Section 4** discusses Pegasys configuration requirements.

- **Section 5** discusses the E-Gov Travel integration application from a transactional standpoint.
- **Section 6** discusses various webMethods administration topics such as log access, startup, and shutdown.
- **Appendix A** discusses information on interface process schedules.

1.2 Document References

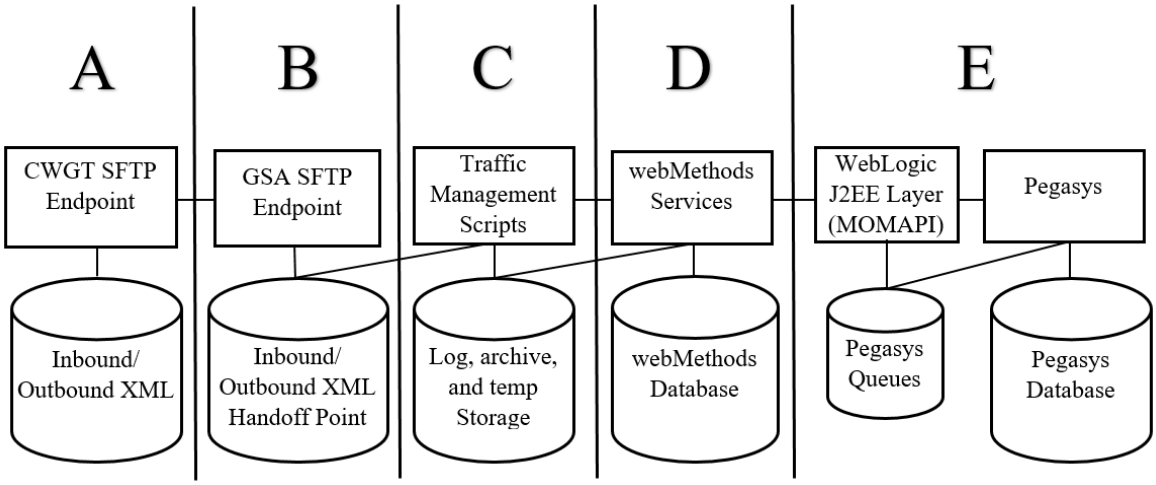
The following references are recommended if additional information about the functional and technical aspects of this integration is required:

- eTravel/Pegasys Interface Integration Agreement
- eTravel/Pegasys Interface Detailed Design Document
- eTravel/Pegasys Interface System Architecture Document
- webMethods Installation Guide
- Momentum MOMAPI Adapter User Guide

2 Secure FTP scripts

This section discusses installation, operation and troubleshooting procedures for the eTravel file transfer scripts. Written in the Perl language, these scripts are responsible for bridging the gap between GSA's SFTP server (section B in **Figure 2**) and GSA's webMethods server (Section D in **Figure 2**). This section of the document contains additional installation and prerequisite information not found in **Sections 3 or 4**.

Figure 2: GSA's webMethods Server



Below are the basic steps in the GSA's webMethods Server diagram in **Figure 2**.

- A
  - CWGT SFTP Endpoint
  - Inbound/Outbound XML
- B
  - GSA SFTP Endpoint
  - Inbound/Outbound XML Handoff Point
- C
  - Traffic Management Scripts
  - Log, archive, and temp Storage
- D
  - webMethods Services
  - webMethods Database
- E
  - WebLogic J2EE Layer (MOMAPI)
  - Pegasys

- Pegasys Queries
- Pegasys Database

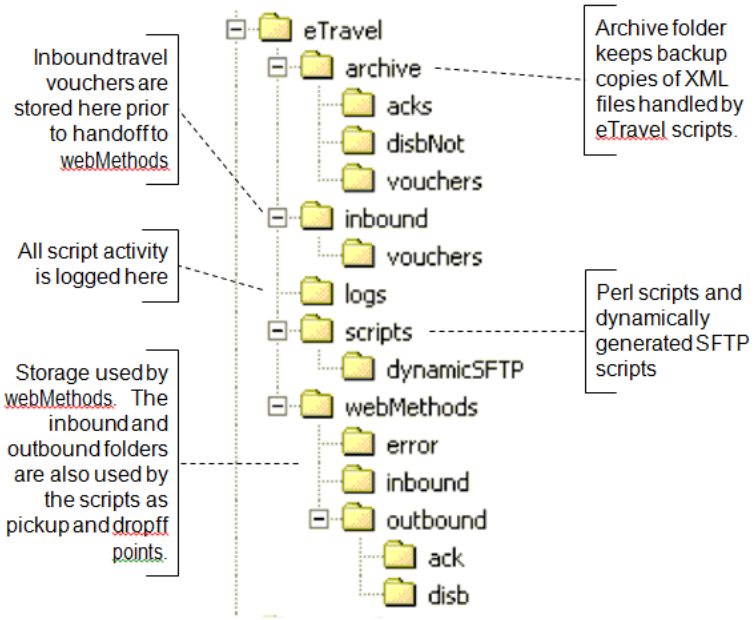
2.1 Prerequisites and Installation

The following software is required for installation and execution of the Perl scripts:

- The eTravel scripts were developed using ActiveState's ActivePerl version 5.8.7. This (or a compatible version) must be installed in order to execute the scripts.
- The scripts use the Mail-Sender package to send email alerts for important events. This is installed using the ppm tool that ships with ActiveState's ActivePerl by typing **ppm install Mail-Sender** at a DOS prompt (the Perl installation directory should be in the path - it is by default). The environment will need access to the internet to retrieve this package.

The scripts and directory structure are contained in a single zip file (eTravel.zip). Once the prerequisite software is in place, eTravel.zip can be unzipped to the root of the C drive. The directory structure looks like **Figure 3**.

Figure 3: Directory Structure



Below are the folders illustrated in the Directory Structure **Figure 3**.

- eTravel
  - archive – Archive folder keeps backup copies of XML filed handled by eTravel scripts
    - acks
    - disbNot

        - vouchers
    - inbound – Inbound travel vouchers are stored here prior to handoff to webMethods
        - vouchers
    - logs – All script activity is logged here
    - scripts – Perl scripts and dynamically generated SFTP scripts
        - dynamicSFTP
    - webMethods – Storage used by webMethods. The inbound and outbound folders are also used by the scripts as pickup and dropoff points
        - error
        - inbound
        - outbound
            - ack
            - disb

Because the Winzip utility does not create zip files with empty folders, a file called placeholder.txt is distributed in all folders that would otherwise have been empty. The placeholder.txt files can be deleted after they are extracted.

The eTravel scripts (located in C:\eTravel\scripts) consist of three files:

- eTravelGSA2CWGT.pl handles the transmission, logging and archiving of disbursement notifications and travel voucher acknowledgements. It periodically checks C:\eTravel\webMethods\outbound\ack and C:\eTravel\webMethods\outbound\disb for outbound transactions.
- eTravelCWGT2GSA.pl retrieves inbound travel vouchers (originating at Concur). The travel vouchers are logged, archived and placed into C:\eTravel\webMethods\inbound for pickup by webMethods.
- eTravelConfig.pl contains a number of configurable parameters and some subroutines that are used by eTravelGSA2CWGT.pl and eTravelCWGT2GSA.pl.

2.2 **Script Operation**

The inbound and outbound scripts execute in the background on AIX, periodically checking for inbound and outbound traffic. The scripts are started manually from a command prompt using the following commands:

Starting the inbound script: /pegasys/software/utils/etravel/scripts/EGovTravInbound.sh

Starting the outbound script: /pegasys/software/utils/etravel/scripts/EGovTravOutbound.sh

2.3 **Script Configuration**

This section details configurable parameters (**Table 1**) found in eTravelConfig.pl.

Table 1: eTravelConfig Parameters

Variable	Usage
$inboundTravelVoucherPath	Inbound travel vouchers retrieved via secure FTP.
$archiveTravelVoucherPath Location to place	Location of archived travel vouchers. XML files are copied here after being placed in the inbound webMethods folder.
$archiveTravelVoucherAckPath	Location of archived travel voucher acknowledgements. XML files are copied here after being transmitted to the secure FTP server.
$archiveDisbNotificationPath	Location of archived disbursement notification XML files. They are copied here after being transmitted to the secure FTP server.
$logFilePath	Log files are maintained here.
$wmInboundXml	New travel vouchers are placed here for processing by webMethods.
$wmOutboundAck	Outbound acknowledgements are placed here by webMethods after attempting to process inbound travel vouchers.
$wmOutboundDisb	Grouped, outbound disbursement notifications are placed here by webMethods.
$wmErrorXml	webMethods uses this folder to store inbound XML that fails to process.
$wmErrorCounterExpected	This is the default number of files in the webMethods error folder. Normally this is 2 (. and ..).
$wmErrorCounter	This is the current number of files in the webMethods error folder. While the outbound script is running, it periodically checks to see if this value is changing, and can optionally send emails.
$rootScriptPath	This is the root directory for the scripts.
$logFilePrefixInbound	Prefix for inbound traffic log file.
$logFilePrefixOutbound	Prefix for outbound traffic log file.
$sendEmails	Flag to indicate whether emails should be sent (0=no, 1=yes)

Variable	Usage
$exceptionEmailRecipientsTo	Comma separated email addresses for “To” recipients.
$exceptionEmailRecipientsCc	Comma separated email addresses for “Cc” recipients.
$smtpFromAddress	Value to use in the From field for E-Gov Travel related emails.
$smtpServer	SMTP server identifier.
$lockOnFailure	This flag can be used to halt script activity on severe errors. When set, scripts will continue to run, but nothing will be transferred until the lock file is cleared (0=Off, 1=On).
$xmlSuffix	Suffix for XML files.

Figure 4 provides a set of sample values based on the E-Gov Travel system testing environment.

Figure 4: Sample Values

```
$inboundTravelVoucherPath = "/pegasys/software/utils/etravel/inbound/vouchers/GSA/";
$archiveTravelVoucherPath = "/pegasys/software/utils/etravel/archive/GSA/vouchers/";
$archiveTravelVoucherAckPath = "/pegasys/software/utils/etravel/archive/GSA/acks/";
$archiveDisbNotificationPath =
"/pegasys/software/utils/etravel/archive/GSA/disbnot/";
$archiveObligationPath = "/pegasys/software/utils/etravel/archive/GSA/oblig/";
$logFilePath = "/pegasys/log/acqapp/etravel/logs/GSA/";
$rptFilePath = "/pegasys/log/acqapp/etravel/reports/GSA/";
$rptFilePrefix = "eTravel Report ";
$rptLocalDocType = "6L";
$rptTDYDocType = "6C";
$wmInboundXml = "/pegasys/software/utils/etravel/webmethods/inbound/";
$wmOutboundAck = "/pegasys/software/utils/etravel/webmethods/outbound/gsa/ack/";
$wmOutboundBadAck = "/pegasys/software/utils/etravel/webmethods/outbound/gsa/bad-
ack/";
$wmOutboundSuppressAck =
"/pegasys/software/utils/etravel/webmethods/outbound/gsa/suppress-ack/";
$wmOutboundDisb = "/pegasys/software/utils/etravel/webmethods/outbound/gsa/disb/";
$wmErrorXml = "/pegasys/software/utils/etravel/webmethods/error/";
$wmErrorCounterExpected = 2;
$wmErrorCounter = 0;
$rootScriptPath = "/pegasys/software/utils/etravel/scripts/";
$logFilePrefixInbound = "eTravel Inbound ";
$logFilePrefixOutbound = "eTravel Outbound ";
$sendEmails = 1;
$exceptionEmailRecipientsTo = 'pablo.foronda@cgifederal.com';
$exceptionEmailRecipientsCc = 'ram.kavala@cgifederal.com';
$exceptionEmailRecipientsInternalTo = 'ram.kavala@cgifederal.com';
$exceptionEmailRecipientsInternalCc = 'pablo.foronda@cgifederal.com';
$reportEmailRecipientsTo = 'le.lam@gsa.gov';
$reportEmailRecipientsCc = 'kevin.young@gsa.gov';
$monitorEmailRecipientsTo = 'geoffrey.schutta@cgifederal.com';
$monitorEmailRecipientsCc = 'todd.albrigo@cgifederal.com';
$smtpFromAddress='eTravelScripts.GSA.Prod@cgifederal.com';
$smtpReportsFromAddress='eTravel.Report.GSA.@cgifederal.com';
$smtpServer = "159.142.1.100";
$lockOnFailure = 1;
$lockFileName = "lock_GSA";
$xmlSuffix = "XML";
$trgSuffix = "TRG";
$inboundSleep = 300;
$outboundSleep = 300;
$remoteVoucherPickupPoint = "/gsa/ext/eTravel/prod/outgoing/vouchers/";
$remoteObligationArchivePoint = "/gsa/ext/eTravel/prod/outgoing/obligations/";
$remoteAckDropoffPoint     = "/gsa/ext/eTravel/prod/incoming/statusupdate/";
$remoteDisbDropoffPoint    = "/gsa/ext/eTravel/prod/incoming/paymentnotification/";
```

2.4 Locking Mechanism

A "lock" file (physically located in $rootScriptPath and called **lock**) is created when critical failures occur (listed below). This can be turned on or off in eTravelConfig.pl via the $lockOnFailure variable (0=Off, 1=On). The inbound and outbound scripts check for the existence of the lock file each time they wake up to perform their respective tasks. If the lock file is found to exist or is created during the operation of the scripts, an email notification indicating its existence is sent. Once the scripts have detected the lock condition, they will take no further action until the lock file is manually deleted (which should not be done until the error condition has been corrected). If the scripts are still executing, they will automatically resume checking for inbound and outbound traffic once the lock file is deleted.

The lock file is created when any of the following conditions are discovered:

- outbound secure FTP script fails to successfully transfer files to secure FTP server
- archival of outbound file(s) fails
- inbound secure FTP script fails to successfully retrieve files
- inbound secure FTP script fails to successfully clean up previously retrieved files on secure FTP server
- inbound script fails to copy travel voucher XML files to webMethods inbound folder
- archival of inbound file(s) fails

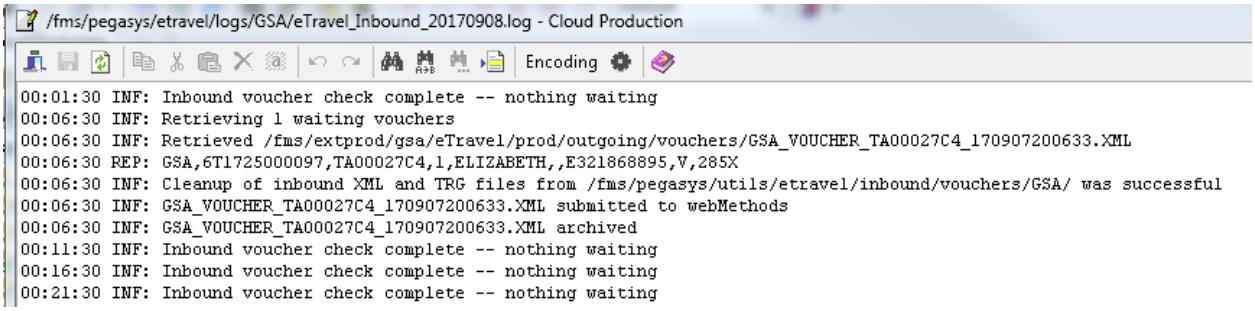
These conditions also cause one email notification to be sent to configured recipients.

2.5 Script Logs

This section shows some examples of inbound and outbound script contents. All script messages begin with a timestamp. The date is part of the log file name, and new log files are created for each day. The log message contents are in plain English, and describe exactly what the scripts are doing. This is a good source of information for troubleshooting as well as documentation for inbound and outbound activity.

Figure 5 shows inbound script example (success).

Figure 5: Inbound Script Success



The Inbound Script Success log in **Figure 5** a plain text file with a row of buttons across the top allowing the user to interact with the content such as cut, paste, search and save. The messages/errors are individual lines of information that each start with a timestamp and states every step of the scripts' process as it completes successfully. For example:

00:01:30 INF: Inbound voucher check complete – nothing waiting
00:06:30 INF: Retrieving 1 waiting vouchers

Figure 6 shows inbound script example (failure).

Figure 6: Inbound Script Failure

```
10:16:42 INF: Inbound voucher check complete -- nothing waiting
10:21:44 INF: Retrieving 1 waiting vouchers
10:22:55 ERR: A problem occurred copying inbound files from pickup
point! 10:22:55 ERR: All files will remain on inbound server.
                Possible causes: 10:22:55 ERR:          1) Destination
drive (/pegasys/software/utils/etravel/inbound/vouchers/GSA\) full
10:22:55 ERR:   2) Trigger file may be present without corresponding XML file.
10:22:55 ERR: Possible failure handling ->
GSA VOUCHER 714441 100120075957784.XML
```

The Inbound Script Failure log in **Figure 6** is a plain text file. The messages/errors are individual lines of information that each start with a timestamp and states every step of the scripts' process as it completes successfully. For example:

10:16:42 INF: Inbound voucher check complete – nothing waiting
10:21:44 INF: Retrieving 1 waiting vouchers
10:22:55 ERR: A problem occurred copying inbound files from pickup point!

Figure 7 shows outbound script example (success).

Figure 7: Outbound Script Success

```
15:04:16 INF: Found 9 acknowledgements and 0 payment files
15:04:16 INF: ACK GSA_STATUS_6C201012277154210_110127030152.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201012277154210,P,Success
15:04:16 INF: ACK GSA_STATUS_6C201101206067510_110127030152.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201101206067510,P,Success
15:04:16 INF: ACK GSA_STATUS_6C201101247674310_110127030152.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201101247674310,P,Success
15:04:16 INF: ACK GSA_STATUS_6C201101268716010_110127030152.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201101268716010,P,Success
15:04:16 INF: ACK GSA_STATUS_6C201101278936410_110127030153.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201101278936410,P,Success
15:04:16 INF: ACK GSA_STATUS_6C201101279025910_110127030220.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6C201101279025910,P,Success
15:04:16 INF: ACK GSA_STATUS_6L201101242524300_110127030140.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6L201101242524300,F,Errors encountered - see Claim History. The following problems were
found: (Third Party
Payment\Third Party Payment Line 1) PC0188E - The transaction date must be within the External Direct
agreement - 3681109 star
t date and billing end date.
15:04:16 INF: ACK GSA_STATUS_6L201101263213200_110127030220.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6L201101263213200,P,Success
15:04:16 INF: ACK GSA_STATUS_6L201101263437200_110127030153.XML copied to
/gsa/ext/eTravel/prod/incoming/statusupdate/
15:04:16 REP: GSA,6L201101263437200,P,Success
15:04:16 INF: Outbound payment and acnowledgement transmission complete
```

The Outbound Script Success log in **Figure 7** is a plain text file. The messages/errors are individual lines of information that each start with a timestamp and states every step of the scripts' process as it completes successfully. For example:

15:04:16 INF: Found 9 acknowledgements and 0 payment files

Figure 8 shows outbound script example (failure).

Figure 8: Outbound Script Failure

```
16:02:37 INF: Outbound payment and acnowledgement transmission complete
16:07:37 INF: Found 3 acknowledgements and 0 payment files
16:07:37 ERR: Failed voucher (file:
/pegasys/software/utils/etravel/webmethods/outbound/gsa/ack/GSA STATUS 6C2010111005483
10 101118040337.XML). Must be manually re-submitted, no ack sent to CWGT.
16:07:37 REP: GSA,6C201011100548310,F,[ART.117.4002] Adapter Runtime (Adapter
Service): Unable to invoke adapter service services:processOffline. [ADA.5001.1032]
MOMAPI MFException: - The following problems were found: (Server Error) AS0068E -
AmsUnitOfWork::Reset - Tried to Reset the unit of work without a transaction Rollback
or Commit after flushing at least one object null
```

The Outbound Script Failure log in **Figure 8** is a plain text file. The messages/errors are individual lines of information that each start with a timestamp and states every step of the scripts' process as it completes successfully. For example:

16:02:37 INF: Outbound payment and acknowledgement transmission complete
16:07:37 INF: Found 3 acknowledgements and 0 payment files

Script Troubleshooting:

Here are some guidelines for dealing with script problems.

- Lock file - When the lock file is created, an email notification is sent. The lock notification email is usually preceded by another email that indicates the nature of the problem. Once the problem is cleared, the lock file can be manually deleted. The scripts will automatically proceed from that point.
- If transactions aren't showing up on either end, make sure the inbound and outbound scripts are still executing by the Windows Services dialog.

3 webMethods Installation and Configuration

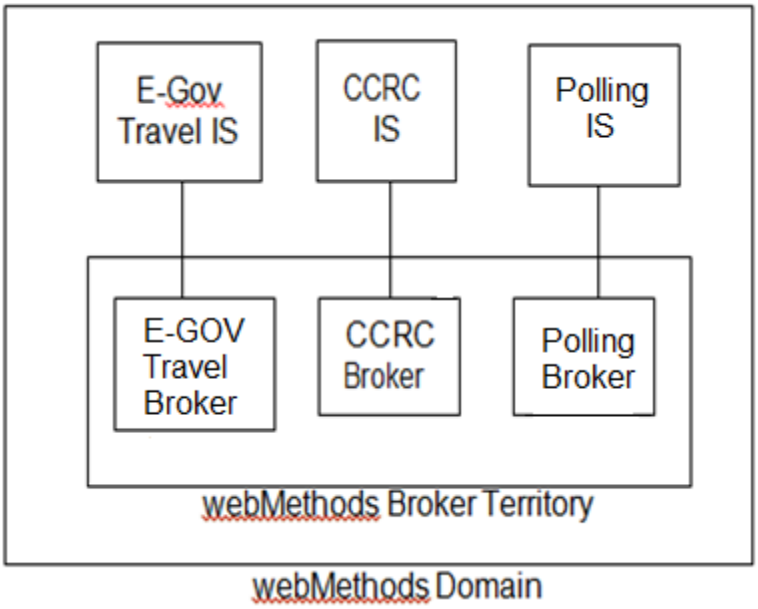
This section covers E-Gov Travel specific configuration items. This document assumes that webMethods has been installed and configured for the environment in which the E-Gov Travel integration software will be deployed. From that point, the following steps are required to install and configure the E-Gov Travel specific instance:

- Install E-Gov Travel specific packages on the designated integration server following instructions found in the webMethods Installation Guide, Prerequisites and Installation Prerequisites and Installation. The packages required by this implementation are:
- AmsMOMAPIAdapter.zip - Enterprise adapter code for Pegasys
- MFAdapterConnection.zip (2.0) - Used to establish MOMAPI connection from adapter to Pegasys
- MfBase.zip (1.0) - Base documents and services
- E2Travel.zip (1.0) - Services to send and receive E-Gov Travel related documents
- MfTravel.zip (1.0) - Service to submit third-party payment documents to Pegasys for processing
- MfTravelConnection.zip (1.0) - Used to establish JDBC connection from adapter to Pegasys
- Synchronize documents (between webMethods Broker and Integration Server). Specific instructions for this can be found in the webMethods Installation Guide, Section 2.9.5.
- Configure and enable the MOMAPI adapter connection following instructions found in the webMethods Installation Guide.
- Configure and enable the JDBC adapter connections following instructions found in the webMethods Installation Guide.
- Configure and enable JDBC adapter notifications following instructions found in the webMethods Installation Guide.

3.1 **Broker Territories**

If the E-Gov Travel integration is to be used in conjunction with other webMethods integrations that rely on the same instance of Pegasys, then it is essential that they all be configured in a single Broker Territory. A Territory allows one or more Brokers to be managed as a single entity, and to each receive all notifications available through the Pegasys MF_Notification queue mechanism (further discussed in **Section 5**). The Broker Territory section of **Figure 1** is repeated, as illustrated in **Figure 9**.

Figure 9: Broker Territory



As illustrated in **Figure 9**, a polling IS and Broker instance is used to collect MF_Notification messages and redistribute them throughout the territory. Installation of the Polling IS and Broker follow the same steps discussed for installation of the E-Gov Travel instance with the exception of the package list. Only AmsMOMAPIAdapter.zip, MFAdapterConnection.zip and MfBase.zip are required for the polling IS as no application specific information is used here. Once the polling instance is created, the webMethods Installation Guide for further configuration instructions.

4 Pegasys Configuration Requirements

As GSA's financial system, Pegasys is responsible for processing incoming travel vouchers as third-party payment documents. The travel vouchers are received via webMethods, which presents the documents to Pegasys via the Momentum API. Subsequently, webMethods is notified of the status of submitted third-party payment documents (whether they processed successfully or failed). webMethods also receives notification of third-party payment disbursements, which it batches for eventual transmission to ConcurGov.

webMethods receives status and disbursement notification messages through the MOMAPI's ETRAV_NotifIn queue (shown in **Figure 2**). This queue is a configurable feature of the Pegasys software. Using the Pegasys registry file, triggers can be established that generate ETRAV_NotifIn messages when certain persistent events occur (e.g., database insert, update or delete). For this integration, triggers are established for third-party payment documents and lines. Any updates to these objects will result in ETRAV_NotifIn queue messages that are retrieved by webMethods.

To enable this feature, the Pegasys registry file must be updated to contain the following lines:

regputd pegasys.rgy /Momentum/ExternalInterface/Notify QueueNames x,y,z

where x,y,z are the notification queues being used by Pegasys. If the line already exists in the registry file, then simply add ETRAV_NotifIn to the list.

regputd pegasys.rgy /Momentum/ExternalInterface/ETRAV_NotifIn/Notify ClassIds x,y,z

where x,y,z are valid class identifiers (numeric tags representing database tables to Pegasys). If the line already exists in the registry file, then simply add the new ClassIds to the list. For E-Gov Travel, 3517 and 3519 need to be present. They represent the Pegasys third-party payment header and Pegasys third-party payment line, respectively.

NOTE: The regputd command should be substituted for the same command that is used in the rest of the registry file. On UNIX, this may be regput51. Also, pegasys.rgy is the production registry name, this may be something else in the test environments (such as test1.rgy, etc.).

Registry maintenance procedures are outside the scope of this document, but basically, the following steps apply once the change is made:

- Regenerate the new registry file
- Shut down Pegasys
- Copy the new registry into place in both the bin and bin/batch directories
- Restart Pegasys

Pegasys should now be queuing messages anytime third-party payment headers or lines are updated, changed, or deleted.

4.1 Treasury Reconciliation Batch Job Execution ID

The Treasury Reconciliation batch job is responsible for setting the state of Pegasys third-party payment lines to "closed" during disbursements processing. This change in the state of the line signifies that the payment has been successfully disbursed, and serves as the trigger mechanism for webMethods to collect information about the payment for subsequent transmission to ConcurGov. The trigger to webMethods is actually a message in the ETRAV_NotifIn queue. During the disbursements process, third-party payment lines are updated frequently, resulting in a large number of ETRAV_NotifIn queue messages. Because webMethods must manually retrieve a payment line each time it is notified of a change, this could result in a lot of database traffic. However, the ETRAV_NotifIn queue entry does identify the execution ID of the process that initiated that entry. Therefore, the execution ID of the Treasury Reconciliation job should be unique, so that it is possible to configure webMethods to ignore messages queued by other processes. The Treasury Reconciliation batch job execution userid is: runbatchtreasrec.

5 Transaction Walkthrough

This section covers the E-Gov Travel integration application from a transactional standpoint. It contains walkthroughs of a travel voucher, travel voucher acknowledgement and disbursement notifications from end-to-end. This information could be useful in the event troubleshooting is necessary.

5.1 **Travel Vouchers**

Travel vouchers are initiated by a traveler using ConcurGov. Travel voucher approval/confirmation precipitate the following steps:

- Integration processes at Concur create an XML document containing the data elements that make up the travel voucher.
- An automated process at Concur moves the file to GSA's secure FTP server using the SSH protocol.
- This server currently resides in Kansas City. Concur transfers the XML files first, and then creates corresponding .TRG (trigger) files for each XML file (same name with .TRG extension instead of .XML). This is done as a safety mechanism so that any process trying to read the XML files can be assured that they have been completely written.
- A Perl script running on the webMethods integration box at GSA periodically checks the secure FTP server for incoming files. It looks in the designated directory for incoming travel vouchers and checks for the presence of trigger files. If there are any, it retrieves the corresponding XML files and deletes them from the secure FTP server.
- The Perl script then places the incoming files in an inbound folder where they will be picked up by webMethods. A copy of the incoming XML file is archived, and all of this activity is logged.
- webMethods picks up the transaction from its inbound folder, transforms the values into a Pegasys third-party payment form (with accounting lines), and submits it for verification. If verification fails, webMethods forms an XML acknowledgement document for transmission back to Concur indicating failure, including the error messages. If verification is successful, the form is next submitted for processing.
- Once processing is complete, webMethods receives notification through the ETRAV_NotifIn queue that the new third-party payment document has been created. It then forms an XML acknowledgement document for transmission back to Concur indicating the transaction processed successfully.

5.2 **Travel Vouchers Acknowledgements**

Travel voucher acknowledgements start with successful or failed third-party payment processing events. webMethods reports these events back to Concur in the form of XML documents using the following steps:

- webMethods creates an XML document containing the data elements that make up the voucher acknowledgement message.
- webMethods deposits the XML acknowledgement in an outbound folder. A Perl script running on the webMethods integration box at GSA periodically checks the webMethods outbound acknowledgement folder for new trigger files. When new triggers are found, the script transmits the trigger and corresponding XML files to GSA's secure FTP server into a folder designated for travel voucher acknowledgements, moves the original from the webMethods folder to an archive area, removes the trigger file, and logs all activity.
- Concur will eventually pickup the XML acknowledgement file and process the acknowledgement.

5.3 **Disbursement Notifications**

Disbursement notification messages are the result of third-party payments being disbursed through the Pegasys Automated Disbursements subsystem. The following steps make up the disbursement notification process:

- The Treasury Reconciliation batch job updates the third-party payment accounting lines to set the state of the line to closed.
- The update triggers an ETRAV_NotifIn queue entry that is processed by the E-Gov Travel webMethods integration server.
- The IS retrieves the third-party payment line and stores some of the attributes in a database table.
- Periodically, the records in the table are grouped into one batch and placed in a single XML file for transmission to Concur.
- The XML file is written into a designated webMethods outbound disbursement notification folder.
- A Perl script periodically checks the outbound webMethods location for XML files. When the disbursement notification file is found, it is transferred to GSA's secure FTP server in a designated location for pickup by Concur. It is then archived and logged.

6 webMethods Administration

This section presents various webMethods administration topics such as log access, startup, and shutdown.

6.1 Logs

There are various logs that hold information for webMethods. Logs can be viewed through the webMethods web interface or on the machine where it is installed. To view the logs on the server, navigate to webMethods\IntegrationServer\logs directory. Logs are formatted server<date>.log, error<date>.log.

- Server.log: startup/shutdown information and service execution information - this is the main webMethods log.
- Error.log: system and application errors.
- Audit.log: system and application audit information.
- Session.log: session information that uses webMethods.
- Repo.log: repository information.
- Stats.log: resource statistics for webMethods.

6.2 Starting the webMethods server

To start the webMethods server, follow these steps:

- Open command prompt on the webMethods machine.
- Navigate to <WMHOME>/IntegrationServer/bin.
- Run server.exe or server.bat.

6.3 Shutting down the webMethods server

To stop the webMethods server please follow these steps:

- Open Google Chrome.
- Use these URLs:
    - http://172.22.202.10:2001/ to access the login page.
    - http://172.22.202.13:2001/ to access the login page.
- Enter user name and password and select OK.
- Select the link located in the top right corner labeled “Shutdown and Restart”.
- Select the appropriate options in the Shutdown and Restart page and select the “Shut Down” button.

Appendix A - Interface Process Schedules

The processes that comprise the E-Gov Travel interface execute periodically, on the following schedule:

- By default, the Perl scripts run continuously, sleeping for 300 seconds between checks for inbound and outbound activity. This interval is configurable.
- ConcurGov integration interface executes 24x7, once per hour.
- The webMethods process that batches disbursement notifications runs 24x7, every 10 minutes.